Le monorepo

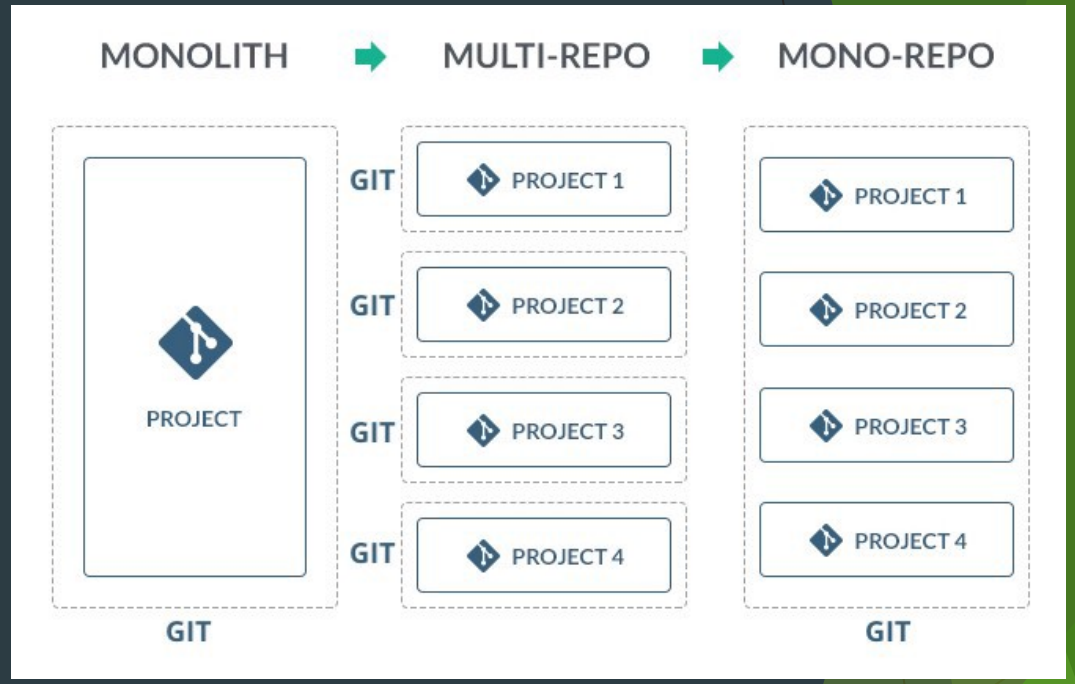
Expliqué par Maxime Ginetti et David Gilson

Monorepos
ça n’est pas

- Une application monolithic
  - Une seule grosse application
  - Tout **builder** en même temps
  - Tout **déployer** en même temps

Monorepos
c’est

- Un unique repository
- Contenant plusieurs projets

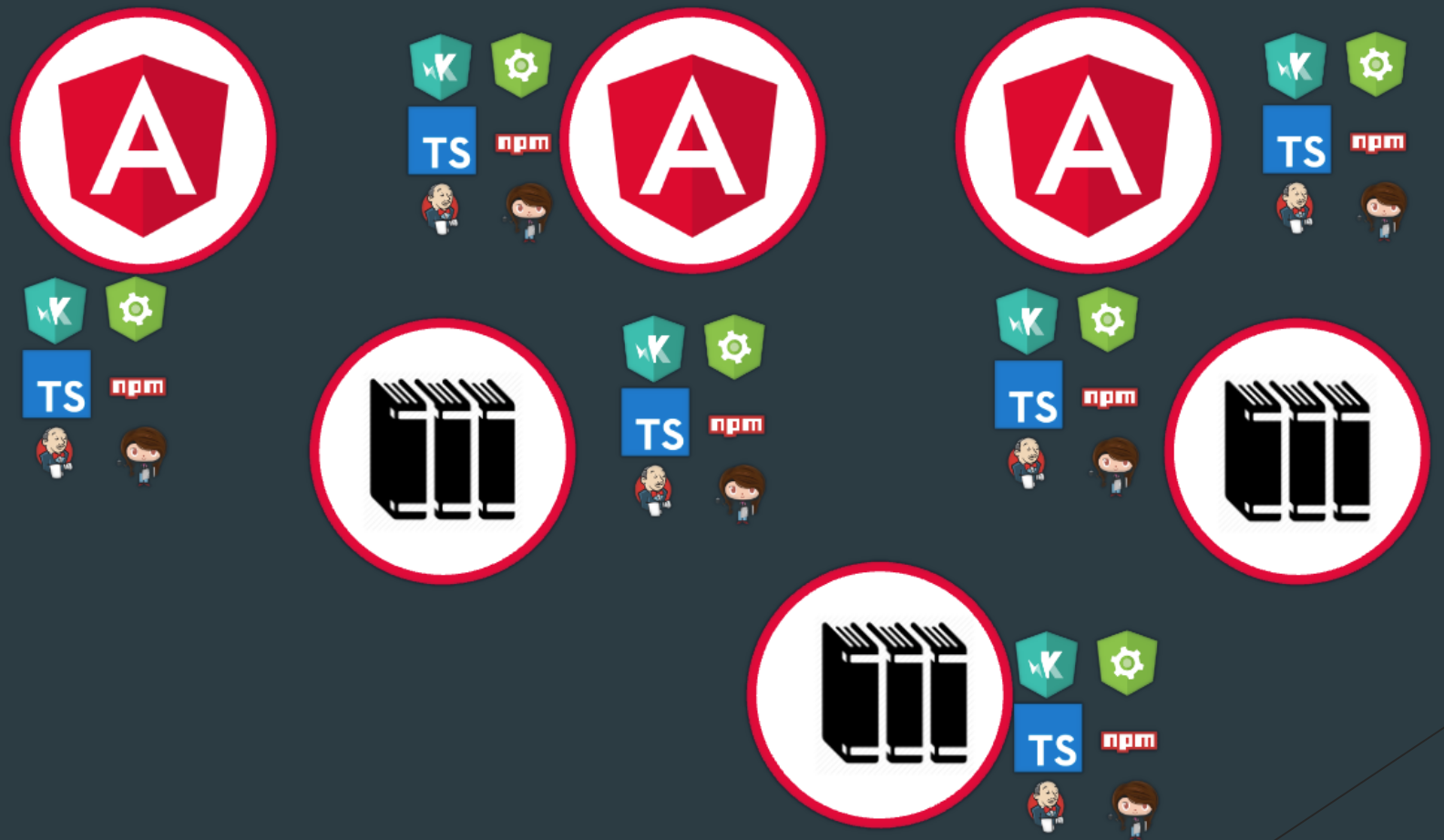


Qui utilise des monorepos ?

- Plus de 500 clients de Nrwl

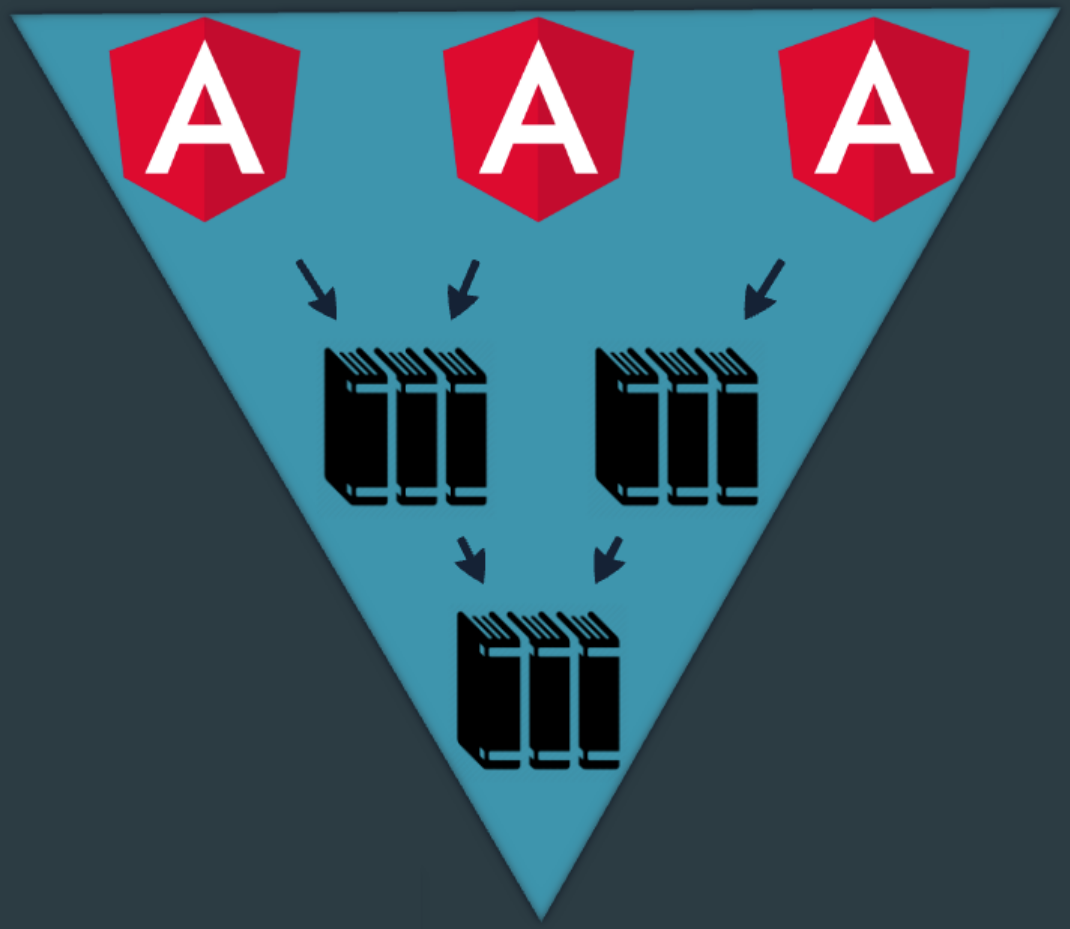
Pourquoi utiliser le monorepo ?

Plusieurs repos



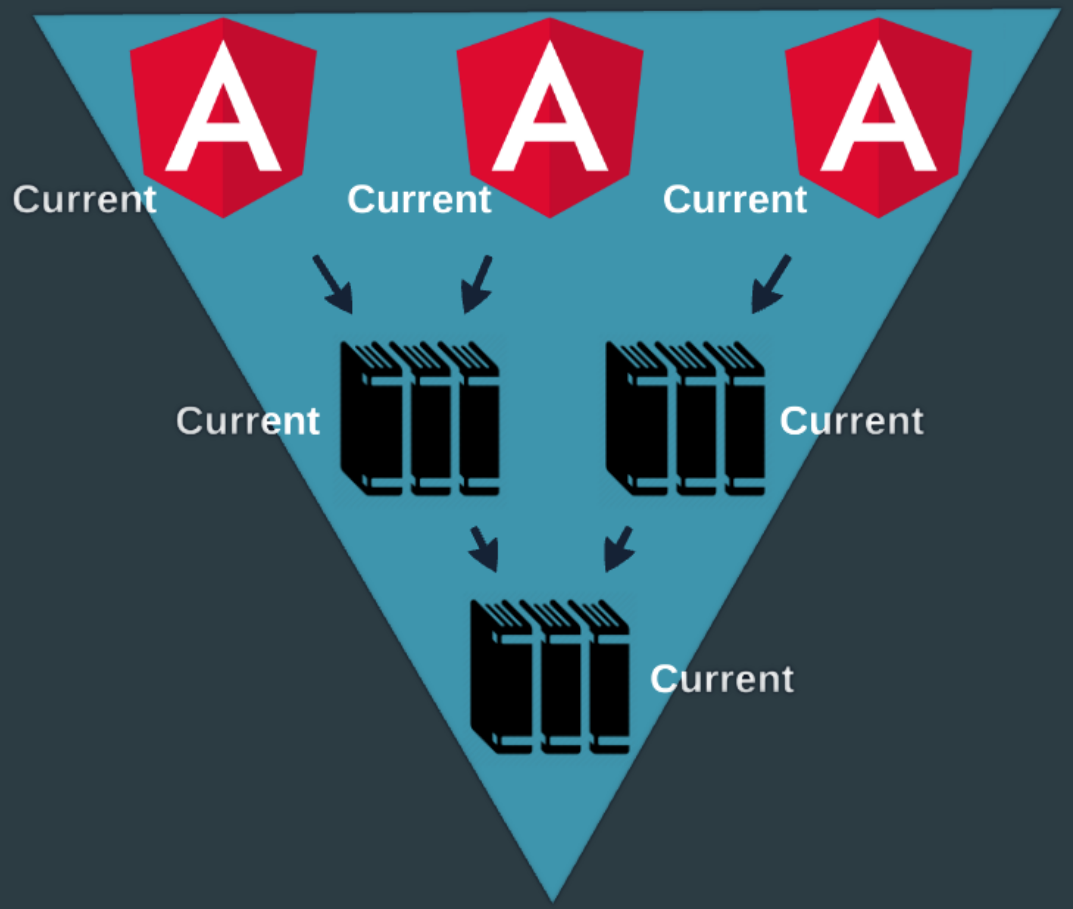
Pourquoi utiliser le monorepo ?

Un seul repo



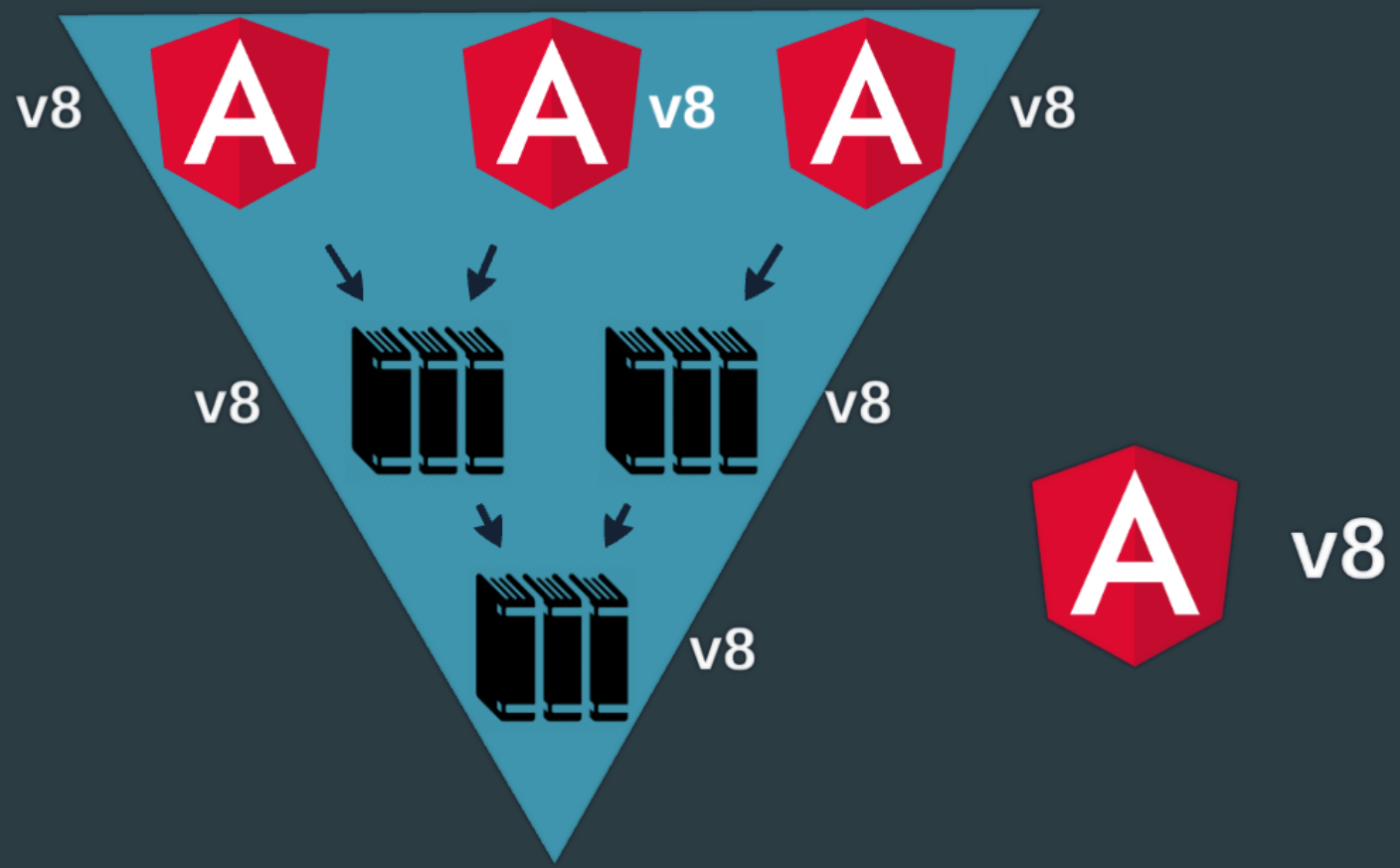
Pourquoi utiliser le monorepo ?

Gestion de version de projets unifiée



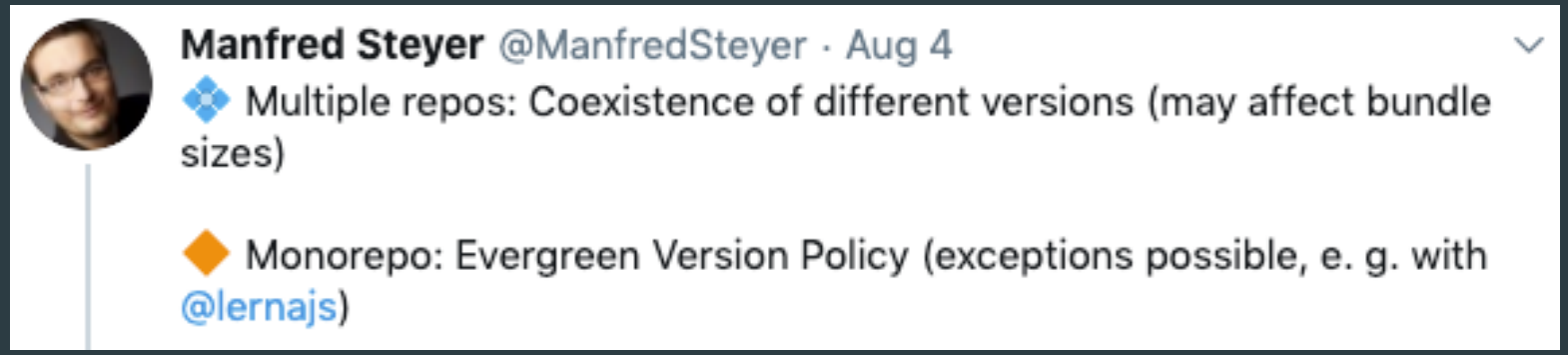
Pourquoi utiliser le monorepo ?

Une seule manipulation d'upgrade Angular

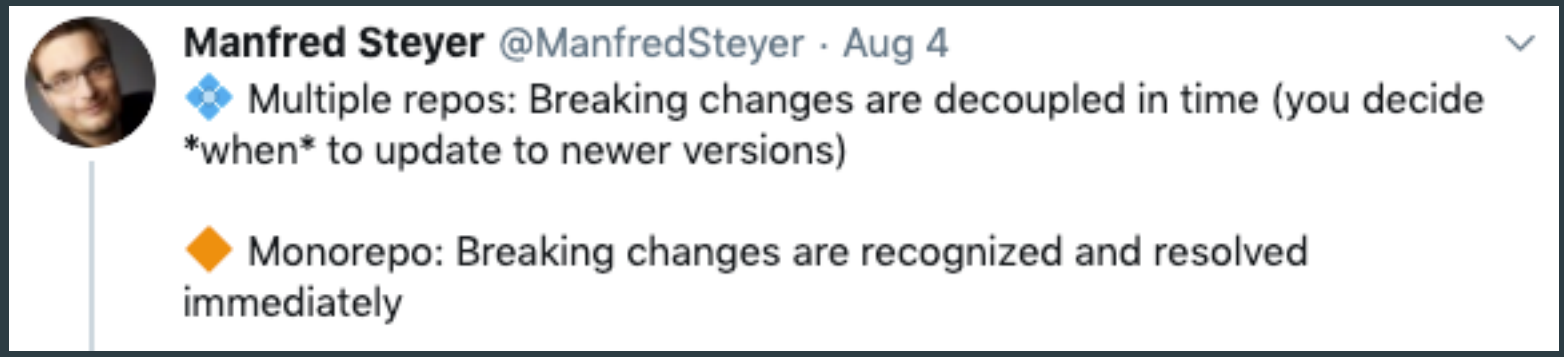


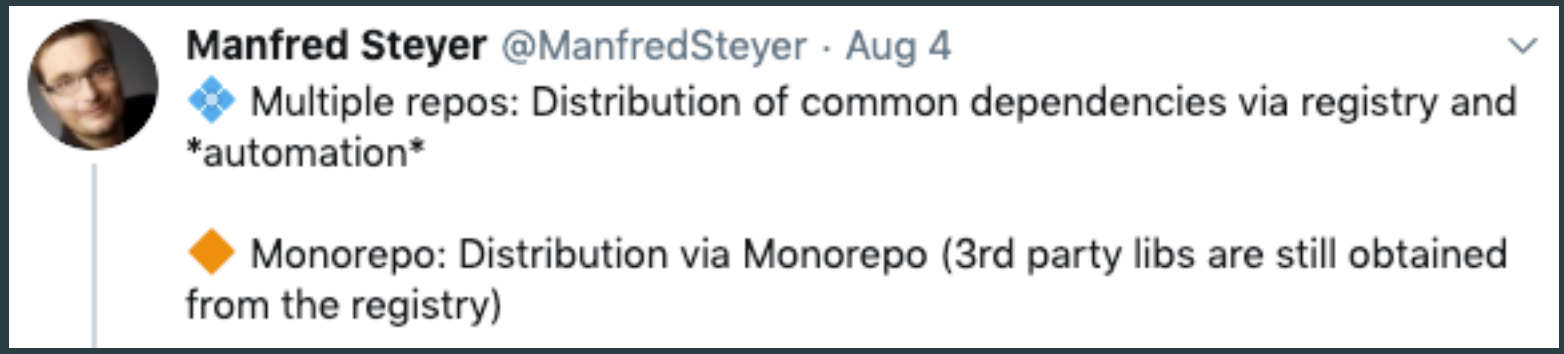
Pros & cons


Manfred Steyer @ManfredSteyer · Aug 4
Replying to @ManfredSteyer
Multiple repos: Maximum flexibility
Monorepo: Intentionally restricting flexibility regarding dependencies

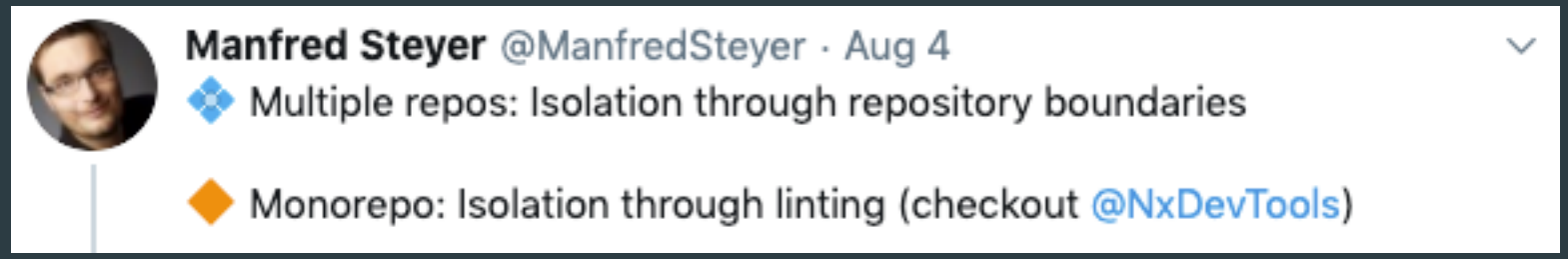

Manfred Steyer @ManfredSteyer · Aug 4
Multiple repos: Coexistence of different versions (may affect bundle sizes)
Monorepo: Evergreen Version Policy (exceptions possible, e. g. with @lernajs)

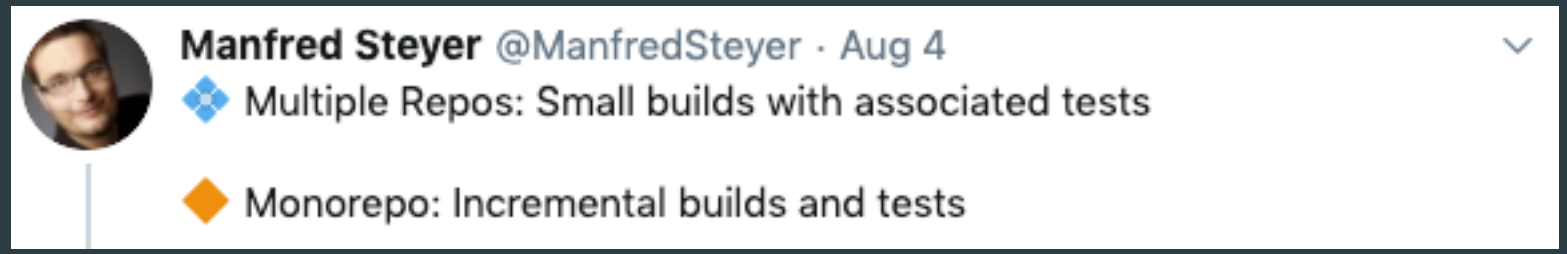
Pros & cons


Manfred Steyer @ManfredSteyer · Aug 4
Multiple repos: Breaking changes are decoupled in time (you decide *when* to update to newer versions)
Monorepo: Breaking changes are recognized and resolved immediately


Manfred Steyer @ManfredSteyer · Aug 4
Multiple repos: Distribution of common dependencies via registry and *automation*
Monorepo: Distribution via Monorepo (3rd party libs are still obtained from the registry)

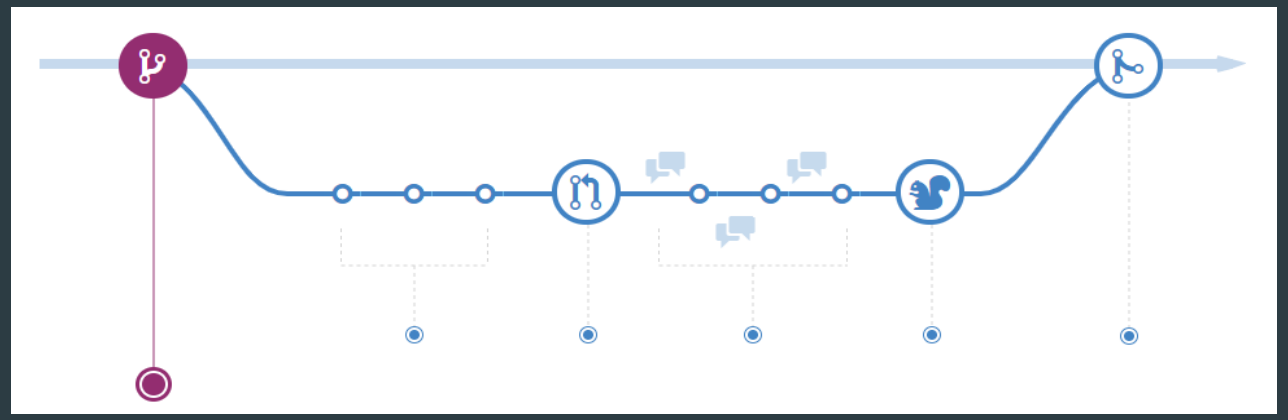
Pros & cons


Manfred Steyer @ManfredSteyer · Aug 4
Multiple repos: Isolation through repository boundaries
Monorepo: Isolation through linting (checkout @NxDevTools)


Manfred Steyer @ManfredSteyer · Aug 4
Multiple Repos: Small builds with associated tests
Monorepo: Incremental builds and tests

Pros & cons

- Git:
  - Bonne gestion des branches et des merges
  - Github flow ? (https://guides.github.com/introduction/flow/)

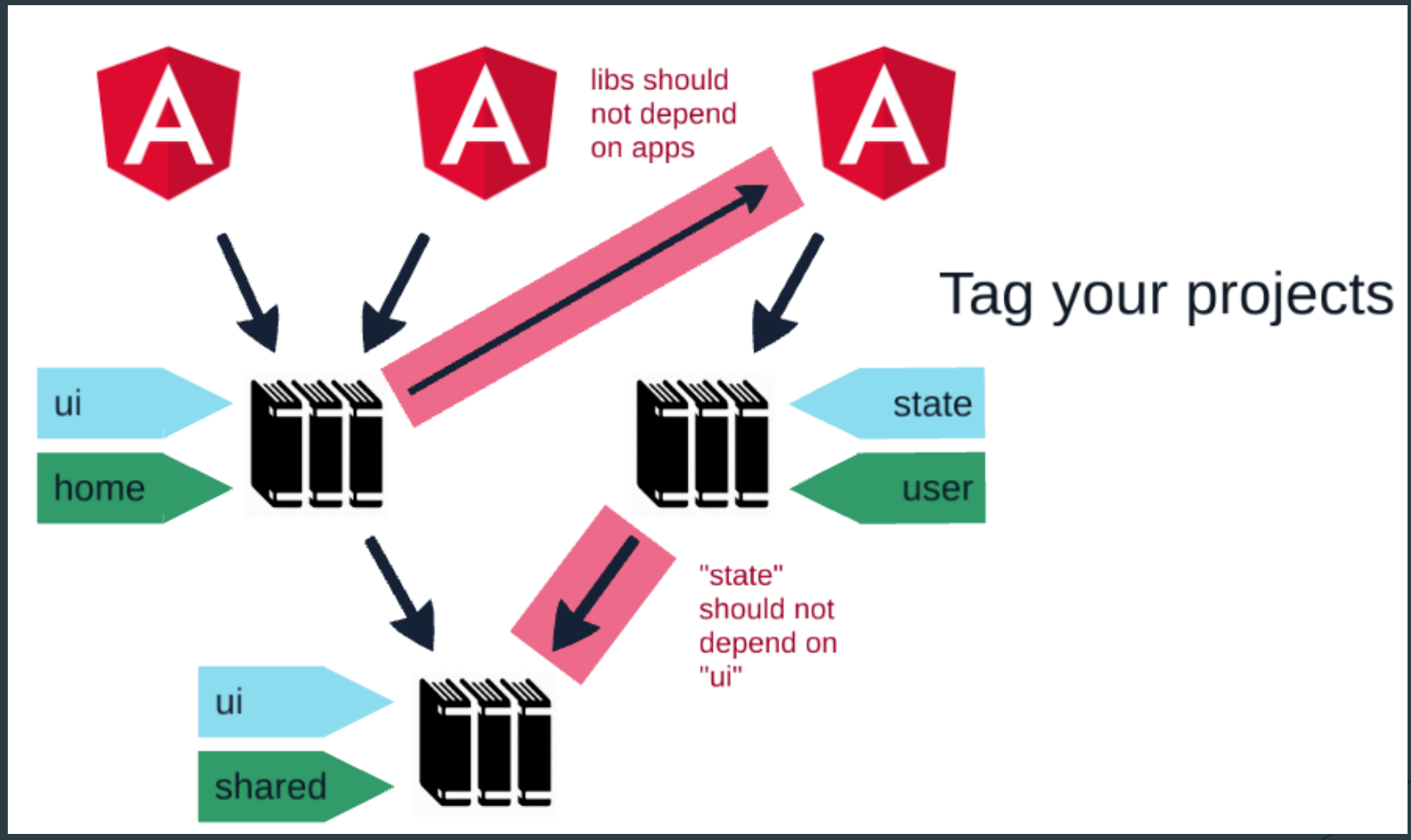


- Pas de restriction d'accès par projet

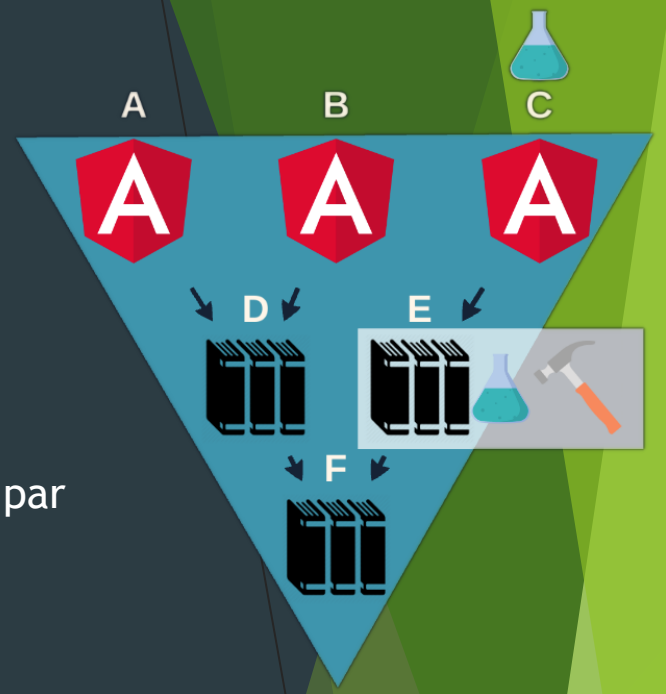
Qu’est-ce que NX ?

- Outil d’aide de gestion pour mono-repo Javascript
  - Un monorepo NX est appelé un Workspace
- Partage de code
  - Facilité de création de librairies réutilisables
- Forcer le respect de standards et de bonne pratiques (de l’entreprise)
  - Création de ces propres schematics de workspace
  - Contraintes d’imports entre librairies/apps
    - Exemple: une librairie Angular UI ne doit pas pouvoir être utilisable dans une app React

Qu’est-ce que NX ?

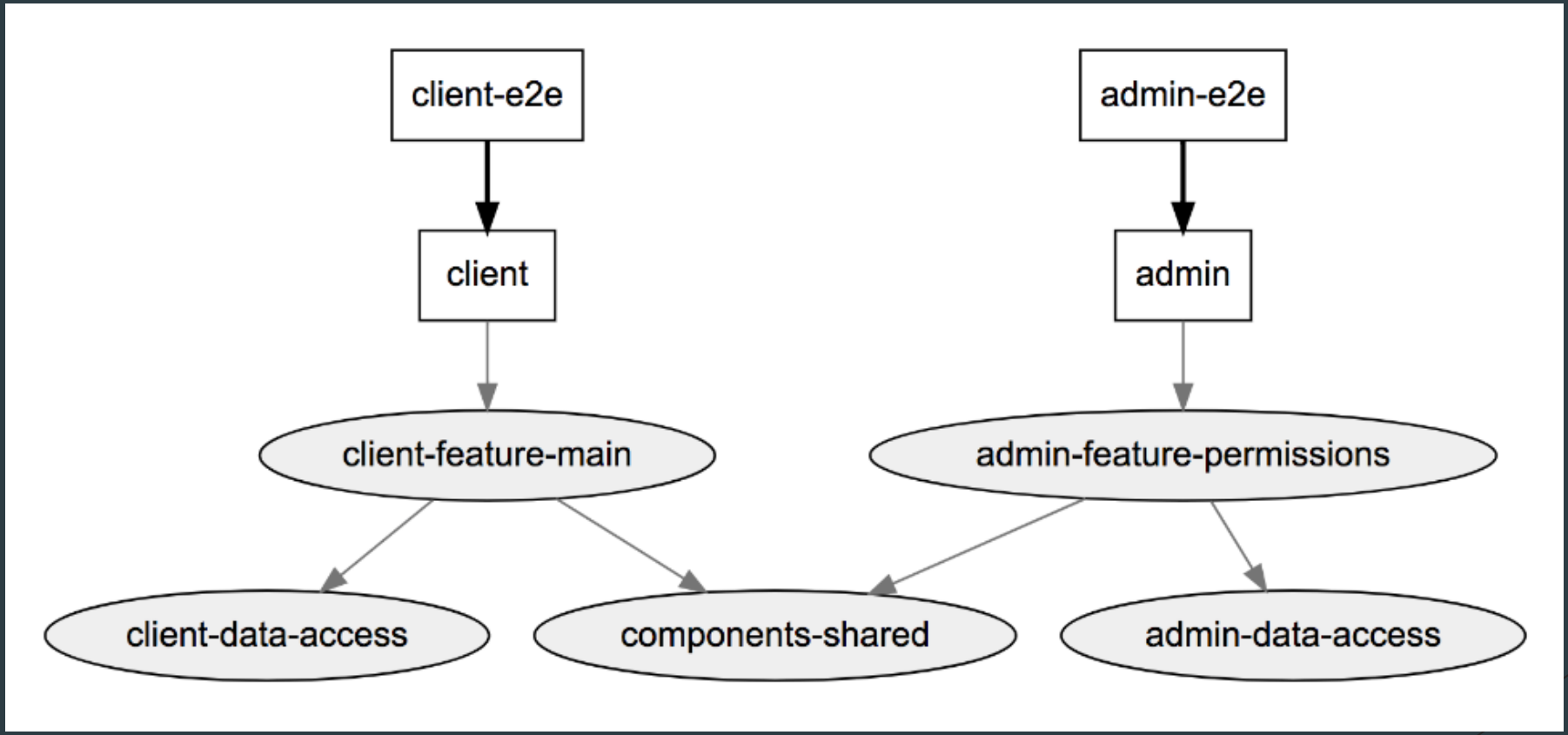


Qu’est-ce que NX ?



- Outils « intelligents » pour l’intégration continue (CI)
  - Appliquer des actions uniquement sur les applications/librairies étant affecté par les changements d’une branche
    - Build
    - Test
    - Lint
- Surcouche du Angular CLI
- Outils de visualisation des dépendances et des impacts de changements

Qu’est-ce que NX ?



Des outils modernes par défaut

- **Typescript**
- **Jest** – Unit testing
- **Storybook** – Développer des composants visuels de façon isolée
- **Cypress** – E2E testing
- **Prettier** – formatage de code

Plugins

- Angular
- NestJs
- Express
- Web (vanilla)
- React
- Next.js
- Gatsby
- Bazel
- Azure
- Storybook

Plugins comunautaires

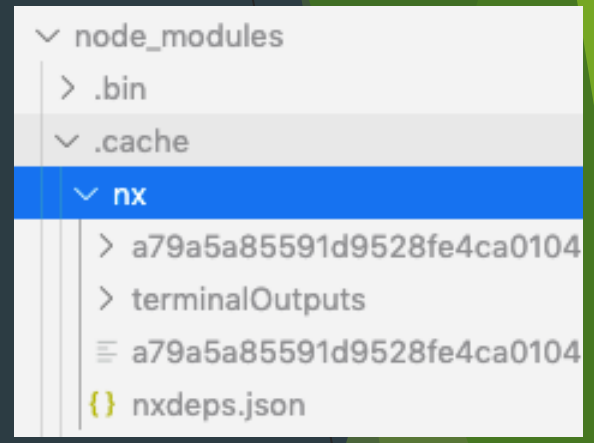
- Capacitor
- Ionic-react
- Domain driven development
- Serverless avec Angular Universal
- Déploiement sur son cloud provider préféré
- ...

Pugins

- Un petit dernier:
  - **nx-plugin**
  - Vous permet de créer votre propre plugin

Caching

- NX a une gestion de cache managée par des hash afin de ne jamais builder 2 fois le même projet « buildable ».
  - Cache local (par défaut)
    - Cache par développeur
  - Cache décentralisé
    - Cache pour la société
    - Cloud
    - On-premise (Alpha) https://github.com/gilsdav/nx-cloud-onprem-runner

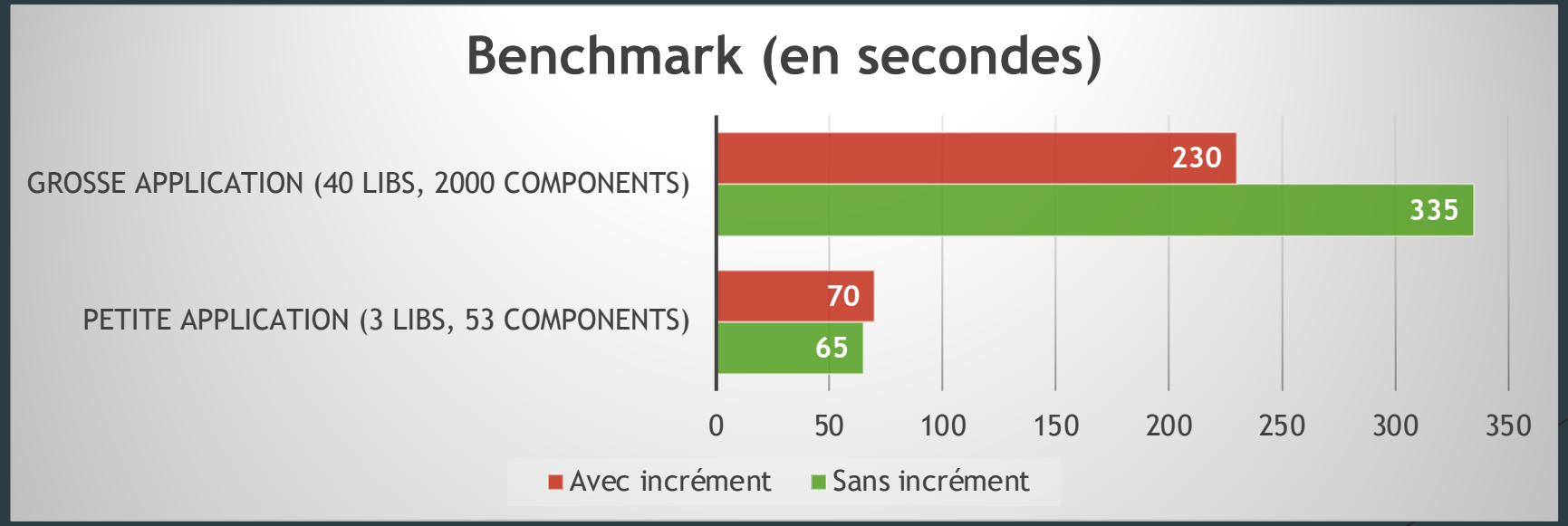


Nx Cloud

Build incrémental

- Utilisation du cache pour éviter de rebuilder les librairies qui n'ont pas changées
- L'application va tout de même empaqueter les librairies déjà buildées

https://github.com/nrwl/nx/issues/3551



Ces benchmarks ont été faits en ne modifiant que l'application, pas une sous librairie. La modification d'une librairie change complètement la donne en pouvant tripler le temps.

Passons au concret

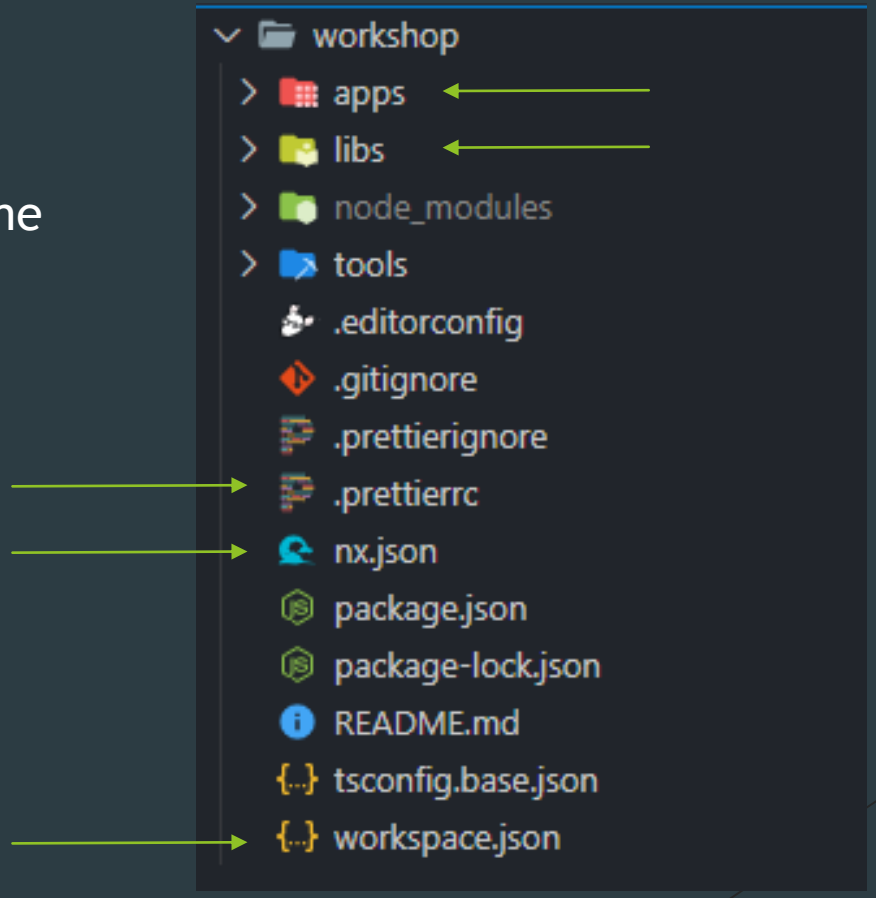
https://github.com/Ginetti/nx-workshop

Workshop Overview

- Création d'un repo NX et d'une application Angular
- Partager une interface de l'app avec un backend NestJS
- Création d'un app React et partage d'un webComponent UI

Création repo nx

- npx create-nx-workspace workspaceName
  - Empty / Nx / No



Ajout de l'app Angular

- npm install --save-dev @nrwl/angular
- npx nx g @nrwl/angular:app angularAppName
  - SASS / No

- Créer une liste de ICard
- https://github.com/Ginetti/nx-workshop/blob/master/apps/myapp/src/styles.scss
  https://github.com/Ginetti/nx-workshop/blob/master/libs/ui-card/src/lib/card/card.component.scss

```
export enum BoxColor {
    BLUE = 'blue',
    GREEN = 'green',
    RED = 'red',
}

export interface ICard {
    title: string;
    boxColor: BoxColor;
    description: string;
}
```

```
<div class="container">
    <div class="header">
        <h1>{{ title }}</h1>
        <img src="https://nx.dev/assets/images/nx-logo.svg" />
    </div>
    <div class="card-wrapper">
        <div *ngFor="let card of cards" class="card">
            <div class="title">{{ card.title }}</div>
            <span class="description">{{ card.description }}</span>
            <span class="box" [ngClass]="card.boxColor"></span>
        </div>
    </div>
</div>
```

Ajout du serveur nestJS

- Partager l'interface et l'enum
  - npm install --save-dev @nrwl/web
  - npx nx g @nrwl/web:lib libName
  - Exporter les fichiers via l'index.ts
- Ajout du serveur
  - npm install --save-dev @nrwl/nest
  - npx nx g @nrwl/nest:app serverName --frontendProject=angularAppName
- Retourner les cartes
  - AppComponent: Get sur l'url http://localhost:4200/serverName/cards

```
@Controller()
export class AppController {
    constructor(private readonly appService: AppService) {}

    @Get('cards')
    getCards() {
        return this.appService.getCards();
    }
}
```

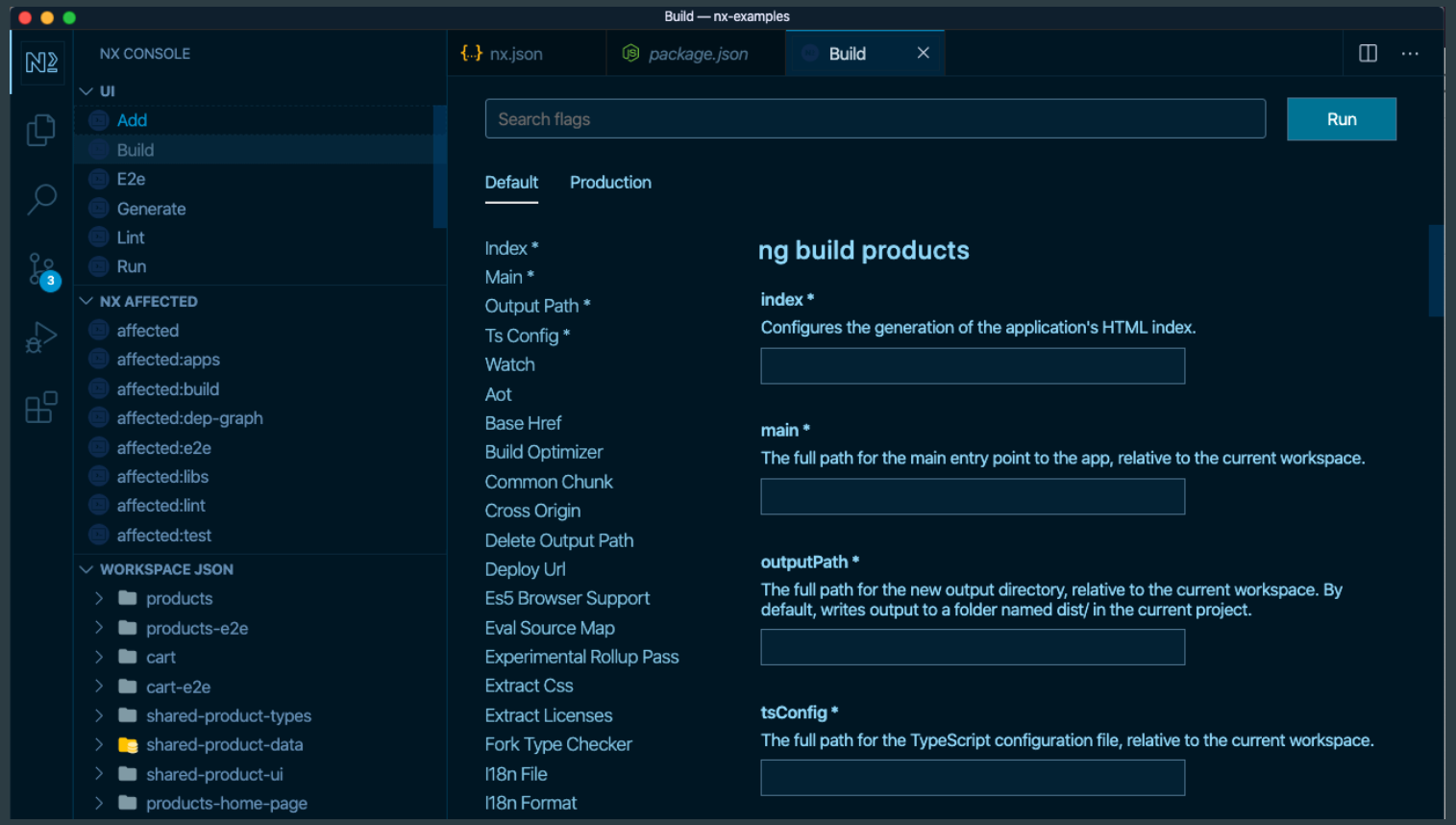
```
@Injectable()
export class AppService {
    getCards(): ICard[] {
        return // Vos cartes
    }
}
```

Ajout app react et partage d'un webComponent

- Ajout du webComponent
  - npx nx g @nrwl/web:lib ui-libName
  - Source: https://github.com/Ginetti/nx-workshop/tree/master/libs/ui-web-card/src/lib
- Adaptation dans l'app Angular
  - Dans le module: `schemas: [CUSTOM_ELEMENTS_SCHEMA],`
  - Dans le component: `import '@projectName/ui-libName;`
  - Source: https://github.com/Ginetti/nx-workshop/blob/master/apps/myapp/src/app/app.component.html
- Ajout react
  - npm install --save-dev @nrwl/react
  - npx nx g @nrwl/react:app reactAppName
  - Configurer proxy dans le workspace.json / angular.json
  - Source: https://github.com/Ginetti/nx-workshop/blob/master/apps/reactapp/src/app/app.tsx

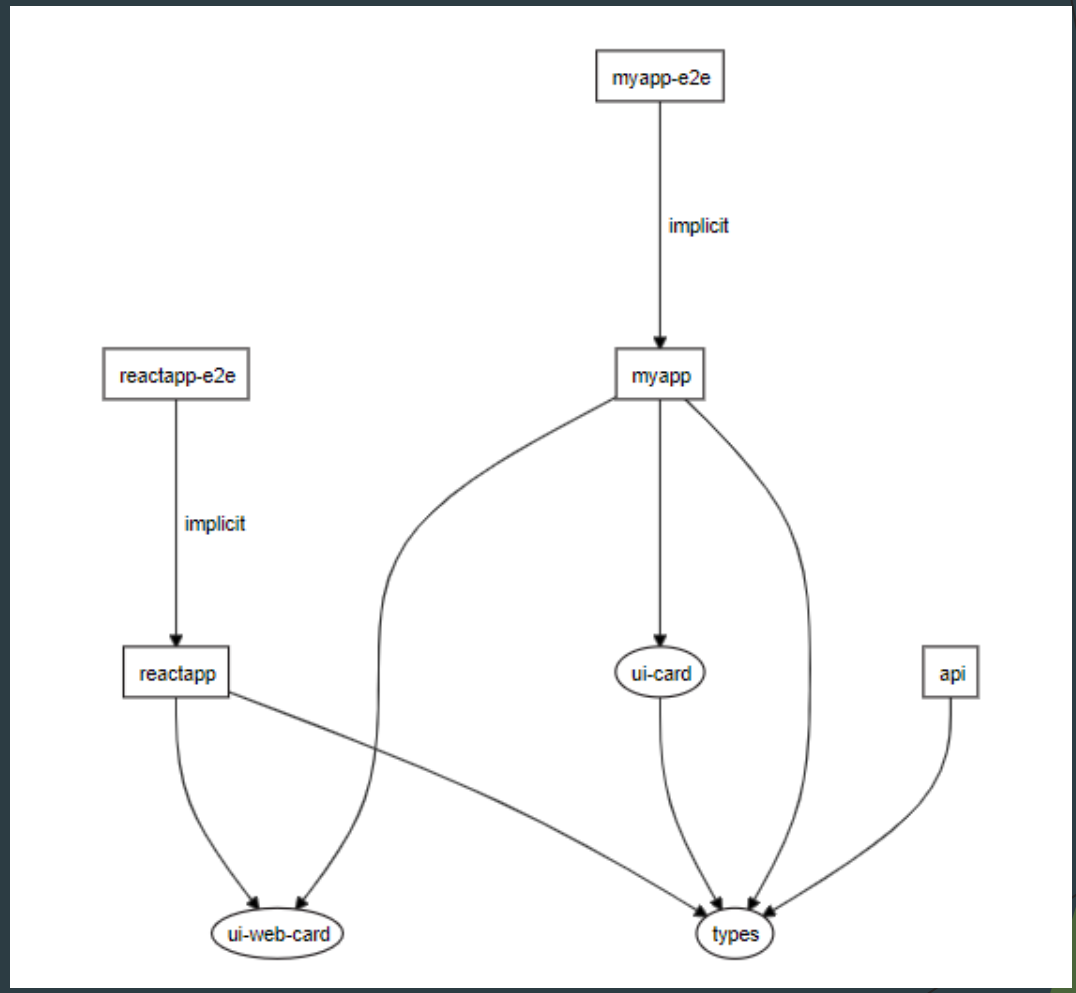
Nx-Console: demo

https://nx.dev/angular/cli/console



Dep-Graph

- npm run **dep-graph**
- npm run **affected:**dep-graph



Nx.json & tsLint

```
"projects": {
    "myapp": {
        "tags": ["scope:myapp"]
    },
    "myapp-e2e": {
        "tags": [],
        "implicitDependencies": ["myapp"]
    },
    "ui-card": {
        "tags": ["scope:myapp", "type:ui"]
    },
    "types": {
        "tags": ["scope:shared", "type:type"]
    },
    "api": {
        "tags": ["scope:api", "type:app"]
    },
    "reactapp": {
        "tags": ["scope:reactapp", "type:app"]
    },
    "reactapp-e2e": {
        "tags": [],
        "implicitDependencies": ["reactapp"]
    },
    "ui-web-card": {
        "tags": ["scope:shared", "type:ui"]
    }
}
```

```
"nx-enforce-module-boundaries": [
    true,
    {
        "enforceBuildableLibDependency": true,
        "allow": [],
        "depConstraints": [
            {
                "sourceTag": "scope:myapp",
                "onlyDependOnLibsWithTags": ["scope:myapp", "scope:shared"]
            },
            {
                "sourceTag": "scope:api",
                "onlyDependOnLibsWithTags": ["scope:myapp", "scope:shared"]
            },
            {
                "sourceTag": "scope:reactapp",
                "onlyDependOnLibsWithTags": ["scope:reactapp", "scope:shared"]
            },
            {
                "sourceTag": "scope:shared",
                "onlyDependOnLibsWithTags": ["scope:shared"]
            },
            {
                "sourceTag": "type:app",
                "onlyDependOnLibsWithTags": ["type:ui", "type:data"]
            },
            {
                "sourceTag": "type:type",
                "onlyDependOnLibsWithTags": []
            },
            {
                "sourceTag": "type:ui",
                "onlyDependOnLibsWithTags": ["type:type"]
            }
        ]
    }
],
```

Advanced

Projet existant

- Transformer le projet en workspace NX
  - **ng add @nrwl/workspace**
  - *Un seul projet (monorepo non compatible)*
- Ajouter la couche de cache mais garder notre projet Angular CLI
  - **npx make-angular-cli-faster**
  - *Un seul projet (monorepo non compatible)*

Jenkins - exemple

```
node {
  withEnv(["HOME=${workspace}"]) {
    docker.image('node:latest').inside('--tmpfs /.config') {

      stage("Prepare") {
        checkout scm
        sh 'npm ci’
      }

      stage("Test") {
        sh 'npm run affected:test --base=origin/master --parallel'
      }

      stage("Lint") {
        sh 'npm run affected:lint --base=origin/master --parallel'
      }

      stage("Build") {
        sh 'npm run affected:build --base=origin/master --parallel'
      }

    }
  }
}
```

Workspace schematics

- Créer: **npx nx g workspace-schematic my-schematic**
- Lancer: **npm run workspace-schematic -- my-scematic …**

- Schematics par défaut d'Angular:
  - @schematics/angular:class
  - @schematics/angular:component
  - @schematics/angular:directive
  - @schematics/angular:module
  - @schematics/angular:pipe
  - @schematics/angular:service

Workspace schematics

```
export default function(schema: any): Rule {

  return chain([

    (tree: Tree, _context: SchematicContext) => {
      ...
      return tree;
    }

  ]);

}
```

Workspace schematics

```
externalSchematic('@schematics/angular', 'module', {
  project: schema.project,
  name: schema.name,
  routing: true,
  module: 'app.module.ts'
}),
```

```
externalSchematic('@schematics/angular', 'service', {
  project: schema.project,
  name: schema.name,
  path: path.join(
    'apps',
    schema.project,
    'src',
    'app',
    schema.name,
    'services'
  )
}),
```

Workspace schematics

```
(tree: Tree, _context: SchematicContext) => {
  const filePath = path.join(
    'apps',
    schema.project,
    'src',
    'app',
    schema.name,
    'hello-world.html'
  );
  tree.create(filePath, `<h1>Hello ${schema.name} </h1>`);
  return tree;
},
```

Workspace schematics

```
function generateFilesFromTemplates(options: any) : Rule {
  return (tree: Tree, _context: SchematicContext) => {

    const sourceTemplates: Source = url(...);
    const destination: string = ...

    const sourceParametrizedTemplates = apply(sourceTemplates, [
      options.spec ? noop() : filter(path => !path.endsWith('.spec.ts')),
      template({
        ...options,
        ...strings
      }),
      move(destination)
    ]);

    const rule = mergeWith(sourceParametrizedTemplates, MergeStrategy.Default);
    return rule(tree, _context);
  }
}
```

Cache

```
"tasksRunnerOptions": {
  "default": {
    "runner": "@nrwl/workspace/tasks-runners/default",
    "options": {
      "cacheableOperations": ["build", "lint", "test", "e2e"]
    }
  }
},
```

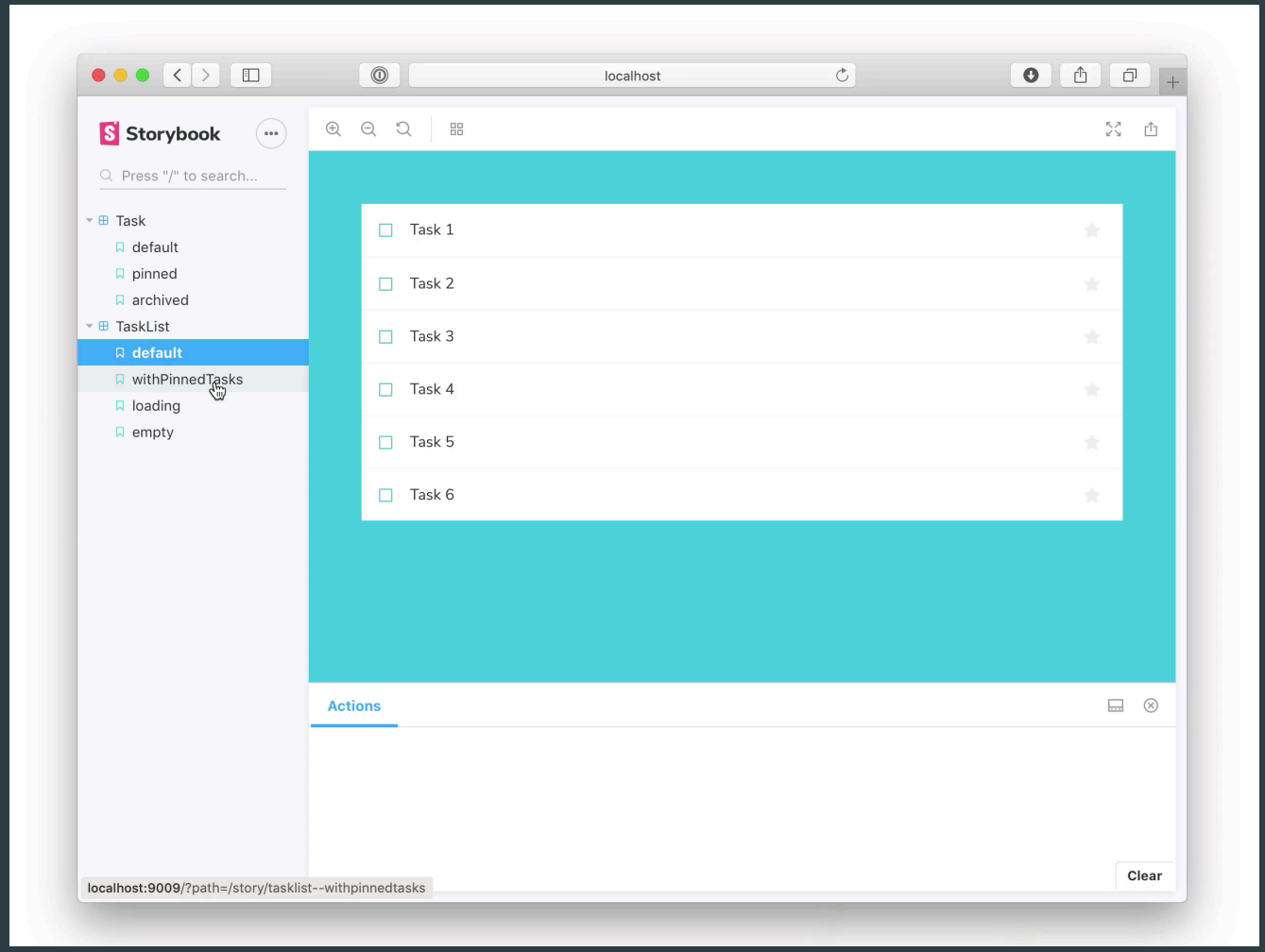
Cache

```
"tasksRunnerOptions": {
  "default": {
    "runner": "@gilsdav/nx-cloud-onprem-runner",
    "options": {
      "cacheableOperations": ["build", "lint", "test", "e2e"],
      "bucket": {
        "url": "http://nx-cache.afelio.be:8080/api"
      }
    }
  }
},
```

Storybook

- npm install –save-dev @nrwl/storybook
- nx g @nrwl/angular:storybook-configuration myapp
- https://www.learnstorybook.com/intro-to-storybook/angular/en/get-started/

Storybook


localhost
Storybook
Press "/" to search...
Task
default
pinned
archived
TaskList
default
withPinnedTasks
loading
empty
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Actions
Clear
localhost:9009/?path=/story/tasklist--withpinnedtasks

Références

- https://nx.dev/angular/cli/overview
- https://nxplaybook.com/p/nx-workspaces
- https://docs.google.com/presentation/d/110_PsQwLwjUcbSNOlJRRdKwwrkCLiG1V4VoVOhkgwT8/edit
- https://dev.to/thisdotmedia/angular-libraries-with-nx-for-enterprise-apps-395h
- https://www.learnstorybook.com/intro-to-storybook/angular/en/get-started/
- https://github.com/Ginetti/nx-workshop
- https://guides.github.com/introduction/flow/